CAD-Based Aerodynamic Design of Complex Configurations Using a Cartesian Method

M. Nemec
NRC Research Associate

M. J. Aftosmis
Senior Research Scientist

T. H. Pulliam
Senior Research Scientist

NASA Ames Research Center
MS T27B, Moffett Field, CA 94035

1 Abstract

A modular framework for aerodynamic optimization of complex geometries is developed. By working directly with a parametric CAD system, complex-geometry models are modified and tessellated in an automatic fashion. The use of a component-based Cartesian method significantly reduces the demands on the CAD system, and also provides for robust and efficient flowfield analysis. The optimization is controlled using either a genetic or quasi–Newton algorithm. Parallel efficiency of the framework is maintained even when subject to limited CAD resources by dynamically re-allocating the processors of the flow solver. Overall, the resulting framework can explore designs incorporating large shape modifications and changes in topology.

2 Introduction

AERODYNAMIC design is inherently a multidisciplinary problem that involves complex surface geometry, competing objectives, multiple operating conditions, and strict design constraints. Consequently, important considerations for an effective optimization framework include: 1) geometry modeling and surface discretization, 2) objective function and constraint evaluation, which includes methods for mesh generation, surface- and volume-mesh perturbation, and flow solution, and 3) the selection of optimization techniques. In modern engineering design environments, the surface geometry is generally represented by a parametric Computer-Aided-Design (CAD) model. Since all downstream analysis and design relies on this representation, the CAD model, accessible in its native environment, should serve as the basis of automated optimization.

Recently, a promising approach has been developed that allows direct access to the native CAD representation. This approach is based on the Computational Analysis and PRogramming Interface (CAPRI) [1, 2, 3, 4, 5]. In addition to providing an effective tool for surface discretization, CAPRI allows the modification of adjustable parameters built into the CAD model. Hence, the design variables and geometric constraints can be intrinsic to the CAD model. Upon a regeneration of the model in response to a parameter change, CAPRI constructs a “water-tight” surface triangulation, which can be automatically refined to obtain a CFD-ready triangulation.

Robust and efficient volume-mesh generation is the next critical part of the optimization framework. Traditional, body-fitted structured and unstructured mesh generation algorithms can be computationally expensive and usually require user supervision. This has motivated the development of mesh-perturbation schemes [6, 7, 8] that are used during the optimization process to modify a given baseline mesh. The location of nodes is tracked as the mesh deforms, which allows the use of fast solution-transfer algorithms and helps maintain a smooth design landscape. Unfortunately, the mesh-perturbation schemes may breakdown and require user intervention for topology and sufficiently large geometry changes.

Cartesian methods offer a promising alternative. The mesh generation is fast, robust, and essentially fully automatic [9, 10]. Due to the decoupling of the surface discretization from the volume mesh, Cartesian mesh generation is virtually insensitive to the complexity of the input geometry. When combined with robust high-fidelity flow solvers, the Cartesian approach provides a unique capability, especially for problems with moving bodies in relative motion [11] and automated optimization [8, 12, 13]. By allowing general topology and radical geometry changes, the optimization algorithm is able to explore new regions of the design landscape that may lead to superior and unconventional designs.

For the problems under consideration here, the most promising optimization algorithms range from autonomous approaches such as evolutionary [14, 15, 16] and finite-difference gradient-based algorithms [17], to methods requiring greater coupling such as the adjoint approach [18, 19, 20] for gradient computations. Furthermore, the use of these techniques in conjunction with pattern-search techniques [21] and approximation methods [22, 23], can help deal with complex design landscapes and reduce the computational cost of the optimization.

The selection of a particular optimizer is problem depen-

dent and involves the classic trade-off between specialization and generality. It is therefore desirable to construct a flexible optimization framework to serve as a test-bed for various strategies and algorithms. Important factors in the integration of an optimization algorithm into such frameworks include: 1) scalability of the optimization technique in a parallel computing environment, 2) degree of coupling among the optimization modules and the high-fidelity solvers within the framework, 3) flexibility in the formulation of objectives and constraints, and 4) effectiveness in multi-modal and noisy design landscapes.

In targeting the design of complex three-dimensional geometry, the presence of noise, or non-smoothness, is unavoidable in the design landscape. The noise stems from three primary sources. First, physical sources such as local flow unsteadiness due to complex geometry may hinder deep convergence of the flow solution. Second, noise due to geometry-representation and discretization, which may be caused by the details of the model construction, the internal characteristics of the CAD system, or the surface tessellation algorithm. Third, noise due to discretization error of the volume mesh. For embedded-boundary Cartesian methods, the intersection of the surface geometry with the volume mesh changes non-smoothly as the geometry evolves during the optimization. Consequently, it is important to evaluate the influence of noise on the optimization algorithm, since the presence of false extrema in the design landscape may slow down and even stall the optimization process.

The objective for this paper is to present the development of an optimization capability for *Cart3D*, a Cartesian inviscid-flow analysis package of Aftosmis *et al.* [10, 24]. We present the construction of a new optimization framework and we focus on the following issues:

- Component-based geometry parameterization approach using parametric-CAD models and CAPRI. A novel geometry server is introduced that addresses the issue of parallel efficiency while only sparingly consuming CAD resources.

- The use of genetic and gradient-based algorithms for three-dimensional aerodynamic design problems. The influence of noise on the optimization methods is studied.

Our goal is to create a responsive and automated framework that efficiently identifies design modifications that result in substantial performance improvements. In addition, we examine the architectural issues associated with the deployment of a CAD-based design approach in a heterogeneous parallel computing environment that contains both CAD workstations and dedicated compute engines. The optimization framework is first validated by solving a lift-constrained drag minimization problem. Thereafter, we demonstrate the effectiveness of the framework for a design problem that features topology changes and complex geometry.

3 Optimization Problem Formulation

The aerodynamic optimization problem consists of determining values of design variables X, such that the objective function $\mathcal{J}$ is minimized

$$\min_X \ \mathcal{J}(X, Q) \tag{1}$$

subject to constraint equations C_j:

$$C_j(X, Q) \leq 0 \qquad j = 1, \ldots, N_{\mathrm{c}} \tag{2}$$

where the vector Q denotes the conservative flowfield variables and N_{c} denotes the number of constraint equations. The flowfield variables are forced to satisfy the governing flowfield equations, $\mathcal{F}$, within a feasible region of the design space Ω:

$$\mathcal{F}(X, Q) = 0 \qquad \forall\ X \in \Omega \tag{3}$$

which implicitly defines $Q = f(X)$. The governing flow equations are the three-dimensional Euler equations of a perfect gas, where the vector $Q = [\rho, \rho u, \rho v, \rho w, \rho E]^T$.

The objective function defines the goals of the optimization problem, while the constraint equations limit the feasible region of the design space. The constraints may involve performance functionals, such as lift, geometric quantities, such as volumes and thicknesses, and also simple bound constraints for design variables. A modular framework is constructed to solve the optimization problem defined by Eqs. 1–3. An evaluation of the objective function and constraints requires the coupling of several software components that form the analysis module of the framework. These components are outlined in Fig. 1 and are described below. Following the analysis module, we present the optimization algorithms and a detailed description of the optimization framework.

4 CAD-Based Geometry Modeling

In traditional approaches for geometry modeling and regeneration, one begins by either importing a given baseline surface discretization to a geometry parameterization tool, or defining a set of idealized components within a geometry parameterization tool [25, 6, 26, 27]. Most likely, the baseline surface discretization has been generated from an existing CAD geometry. This approach offers fast and accurate regeneration of surfaces and computation of component intersections. Furthermore, the source code is usually available, and hence the computation of design sensitivities (if required) is possible by the use of automatic differentiation or analytically.

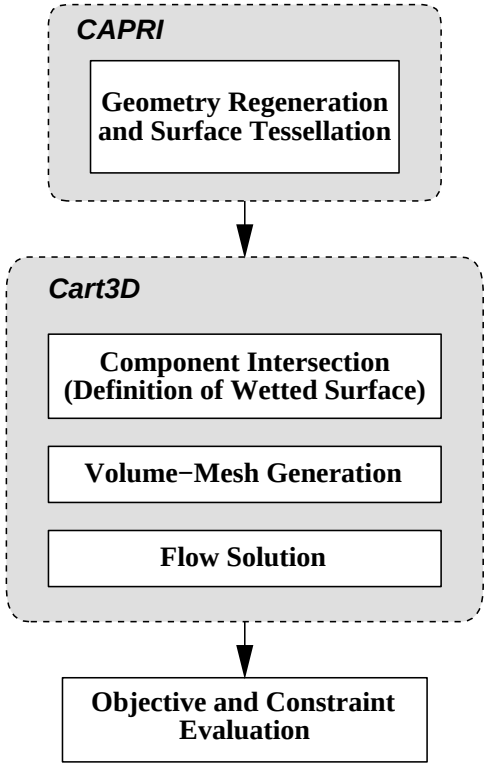


Figure 1: Components of the analysis module

However, the geometry parameterization tool is usually tailored to a specific set of allowable topologies, and provides a limited variety of design variables. For example, only wing-body configurations with prescribed design variables for planform and shape changes may be allowed. Such built-in restrictions limit the feasible region of the design space, and consequently, the best design may never be realized. Although it is always possible to improve the parameterization tool with additional code development, this burden becomes prohibitive when faced with complex, integrated configurations and multidisciplinary problems. Ultimately, this effort leads to the development of a specialized in-house tool mimicking aspects of a parametric CAD system.

An alternative approach is to consider the use of a commercial-CAD system [28]. Most present-day CAD software is based on parametric design and feature modeling. A part is constructed by defining features with adjustable parameters. The features define a sequence of operations, for example an extrusion of a sketched cross section, and are organized in the form of a feature tree. This forms the master-model of the part, and different instances of the part can be generated for various parameter values by following the template of the master-model. Individual parts can be grouped in hierarchical dynamic assemblies, which allow relative motion between constituent parts. Furthermore, features not required for the analysis (or design) problem at hand can be suppressed. The potential advantages of this approach include:

- Generality: there are virtually no pre-defined limitations on the complexity of parts and assemblies.
- Consistency: the part is always queried in its native environment, without geometry translation.
- Variable fidelity: through the use of feature suppression, various levels of part abstraction are possible.
- Natural constraints: the feature-based modeling captures the design intent of the part or assembly and therefore can be used to impose natural constraints on the geometry.

Although this approach is conceptually very appealing, the integration of a commercial CAD system into an optimization framework requires careful consideration of the following issues:

1. Parts and assemblies must be created with design modifications in mind. Although this sounds obvious, the selection of parameters for a design study can be a challenging task and the construction of flexible and robust CAD models requires significant CAD-system experience. The geometry parameterization issue is placed well upstream in the design/ analysis process.
2. The use of “legacy” geometry, or geometry with no parametric CAD representation, requires special consideration. Unfortunately, most CFD geometry today belongs to this category.
3. The interface for accessing the parameters of the CAD model depends on the specific CAD system.
4. The efficiency of the geometry updates and the surface discretization depend on the attributes of the proprietary CAD-geometry kernel.
5. Practical issues such as the number of available CAD licenses need to be considered in the design of parallel optimization procedures.
6. The issue of differentiability and the use of mesh-perturbation algorithms for non-smooth changes in the surface discretization.

Items 1 and 2 are organizational issues that are beyond the scope of this work. It is clear, however, that current production and development environments make extensive use of feature-based solid modeling for engineering analysis and design. While the mesh requirements of CFD simulations place unusual demands on the CAD system, it is highly advantageous to leverage its sophisticated modeling capabilities. We use CAPRI [3, 5, 4] to address items 3 and 4, which

we discuss in the following section. The architecture of the optimization framework, presented thereafter, mitigates the concerns of item 5. Item 6 remains an open issue. Finite-difference schemes can provide good approximations of sensitivity information, but their dependence on stepsize limits the accuracy, and in some cases the robustness, of this approach [26]. Mesh-perturbation schemes introduce additional difficulties due to the requirement of tracking surface deformations [4].

5 Role of CAPRI

5.1 CAD-Model Regeneration

CAPRI exposes the master-model feature tree of the CAD model and allows direct modification of parameters within that tree. A detailed overview of CAPRI's extensive capabilities is given in [5]. An alternative to CAPRI is the direct use of "Developer Toolkits" that are available for most CAD systems. CAPRI, however, provides a unified interface for most CAD systems.

Most design variables are associated directly with values exposed in the feature tree. An exception is surface shape modification, which requires the access to feature information at a high level of detail. For example, the control-point locations for individual curves are required. CAPRI is able to expose non-dimensional curve data points of sketched features, which can be modified to generate new surfaces. For example, these may be the data points of airfoil sections that are lofted to define a wing, or fuselage cross-sections. Note that the use of each data point as a design variable would lead to very large optimization problems and potentially non-smooth curves. To circumvent this difficulty, we use B-spline curves to define each cross-section. The shape design variables are associated with the B-spline control points and are external to the CAD system. This additional level of indirection can easily accommodate other approaches, such as the Hicks-Henne shape functions [4].

Fig. 2 shows an example of two very different instances of the same parametric-CAD model for a generic wing part. The generic model consists of the typical planform parameters that include surface area, aspect ratio, taper ratio, sweep, and root-section and tip-section twist. A shape parameterization example is shown at the top of Fig. 2, where a cubic B-spline with 15 control points is used to closely approximate the RAE-2822 airfoil. The root and tip airfoil sections are linearly lofted to generate the wings shown.

5.2 Automatic Surface Tessellation

After modifying and regenerating the CAD-model, CAPRI provides a surface triangulation for each component. The triangulation is refined based on three measures of quality [3]:

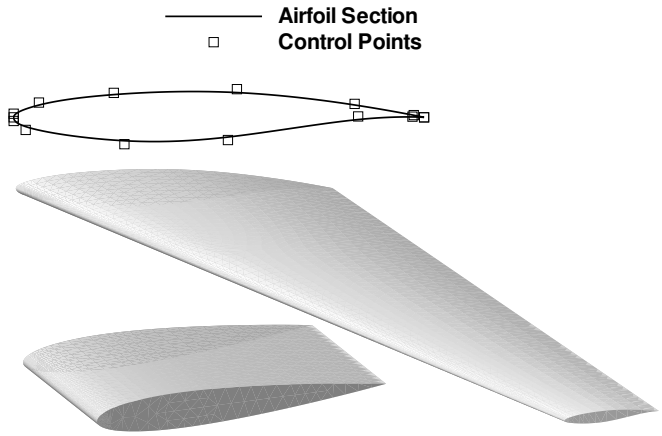


Figure 2: Example of two instances of a generic-wing CAD model. A B-spline airfoil parameterization is shown at the top of the figure.

1) triangle edge length, 2) the deviation of an edge from the underlying CAD model, and 3) a dihedral angle bound between adjacent triangles. The triangulation algorithm is highly robust, but in certain instances the resulting triangulations for a component subject to small shape perturbations may be significantly different. This may introduce noise into the optimization problem, which we discuss further in the Results section.

Recently, a new triangulation algorithm has been added to CAPRI that provides a more uniform, right-triangle based tessellation. An example of coarse triangulations is shown in Fig. 3 for a dramatic change in surface shape. The algorithm identifies component faces that qualify for such triangulations, and otherwise reverts back to the quality triangulation. The new triangulation algorithm is less sensitive to small geometry perturbations.

6 Mesh Generation and Flow Solution

The extraction of a wetted surface from a set of intersecting components is the next task of the analysis module (see Fig. 1). Since CAD-solid representations typically rely on the use of parametric B-splines (or NURBS), the computation of component intersections can be costly within the CAD system. In the present approach, the components are intersected after the surface discretization. This operation is performed efficiently as a part of the component-based approach of *Cart3D* [10]. It should be noted that CAPRI caches an associated triangulation with each component. This caching avoids unnecessary re-triangulations for components that are not modified or experience only rigid body motion during the design process.

Cartesian volume meshes are generated by repeated cell division of an initial coarse mesh [10]. A parallel multi-

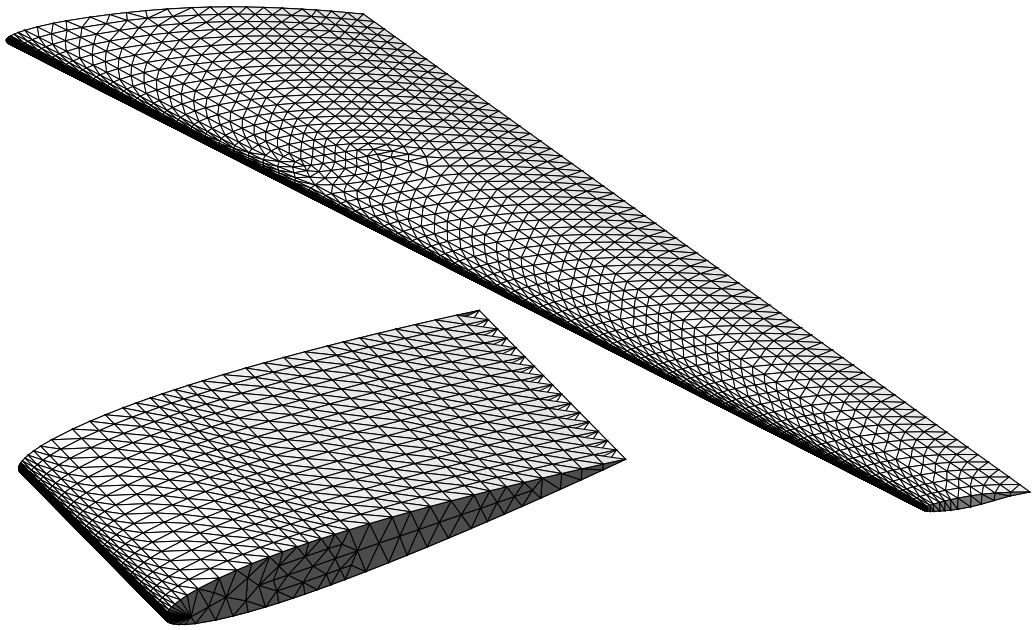


Figure 3: Examples of right-triangle based tessellations for large shape deformations

level method is used to solve the steady-state Euler equations. The spatial discretization is second order accurate using van Leer's flux vector splitting in conjunction with either Minmod or Venkatakrishnan's flux limiters, see Aftosmis *et al.* [24] for details.

7 Optimization Algorithms

We cast the optimization problem as an unconstrained problem by lifting the side constraints, Eq. 2, into the objective function using a penalty method. The constraint imposed by the flowfield equations, Eq. 3, is satisfied at every point within the feasible design space, and consequently these equations do not explicitly appear in the formulation of the optimization problem. We investigate the genetic algorithm of Holst and Pulliam [16], and an unconstrained BFGS quasi-Newton algorithm coupled with a backtracking line search [17, 29, 20]. The objective function gradient is evaluated using central-differences. We "warm-start" the finite-difference gradient computations from the base-state solution, saving roughly 25 to 50% when compared with the standard full-multigrid startup. The solution-transfer algorithm is described by Aftosmis *et al.* [30].

8 Optimization Framework

The synthesis of individual modules into an automated and efficient optimization framework is a challenging software design problem. A number of sophisticated frameworks exist, such as the DAKOTA toolkit [31], which provide a flexible and general approach for linking analysis tools with optimization techniques in large parallel computing environments. In order to have a direct control over the layout of the framework, and therefore quickly evaluate different parallel architectures, we pursue the development of a custom framework.

The first part of the framework addresses the coupling of the CAD/CAPRI module with the optimization process. Figure 4 shows the layout of this distributed client-server interface. The optimization along with the analysis module are executed in a queue system of large compute engines. At each iteration of the optimization process, CAD geometry requests are generated for different parameter values and these are placed in a central repository (right side of Fig. 4). Independent of the optimization runs, a geometry server is initiated that consists of multiple CAD nodes (left side of Fig. 4). The nodes process the geometry requests by retrieving the required parts or assemblies from a specified storage location, regenerating the CAD models, and providing surface triangulations for the optimization processes. Since the geometry requests are independent, we expect the geometry server to achieve nearly linear scalability.

Initially, it may appear that in order to obtain an efficient geometry server, the number of CAD nodes should match the number of geometry requests from all optimization processes. In practice, the number of CAD nodes is limited by the number of available CAD licenses, as each node consumes one license. An immediate concern is that the CAD

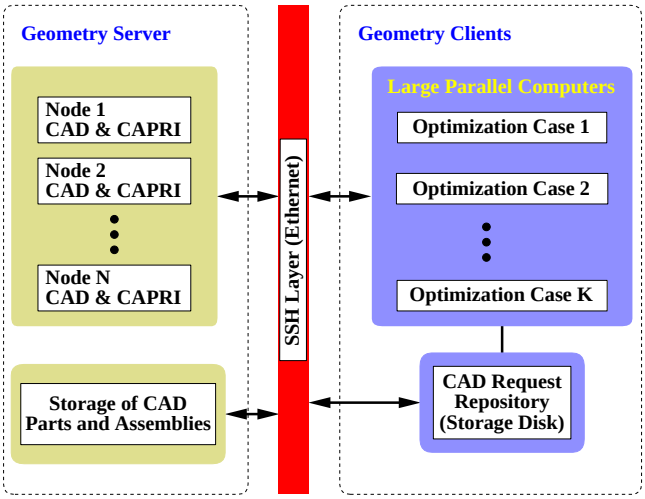


Figure 4: Layout of the interface between optimization processes, or geometry clients, on the right side and the distributed geometry server on the left side.

nodes become the bottleneck of the optimization process, idling the processors of the compute engines. *One of the driving requirements in the design of the present geometry interface is to maintain the efficiency of the optimization process when only a handful of licenses are available, yet remain scalable should the number of licenses increase.*

This is a classic problem of latency. To avoid the geometry processing bottleneck, we mask the latency of the CAD nodes by dynamically allocating the available processors of the optimization process to the number of completed surface triangulations. Figure 5 illustrates this on an example with 64 processors. At the start of each design iteration, all processors are dedicated to the solution of the first returned surface triangulation from the CAD nodes. This is the base state of the gradient method and the first chromosome of the genetic algorithm, denoted as "Geometry 1" in Fig. 5. Note that there is a brief idling of all processors, which could be avoided by implementing an asynchronous optimization approach. Upon completion of the first geometry analysis, we check the number of completed surface triangulations. These are processed by the CAD nodes while the analysis of the first geometry is performed on the compute engine, denoted as "Geometries $2 \ldots K$" in Fig. 5. The number of processors is distributed among the completed surface triangulations and multiple analysis modules are executed on subsets of the available processors. This cycle repeats until all geometry requests are analyzed. For example, the optimization process may have 64 processors available and if 4 surface triangulations are completed by the CAD nodes, we can execute 4 analysis modules in parallel with 16 processors per module, see Fig. 5.

It is important to note that the geometry intersection and volume mesh generation algorithms of the analysis module (see Fig. 1) are serial algorithms, while the flow solver is an efficient parallel solver [24]. The dynamic, coarse-grained parallelism used during each design iteration provides not only concurrent execution of serial tasks, but also ensures high parallel efficiency of the flow solver by limiting the number of processors available to each analysis module. Studies by Eldred *et al.* [32] demonstrate that such multilevel parallelism significantly improves the scalability of optimization frameworks.

The worst case scenario occurs when the wall-clock time required for the processing of a geometry request exceeds the time for completion of the flow solution when all processors are used. If only one CAD node is available, then this CAD node would not be able to feed the compute engine with geometries without processor idle time. This situation is unlikely, since CAD model regeneration and tessellation tasks have computational complexity of $\mathcal{O}(N^2)$, while volume mesh generation and flow solution tasks are $\mathcal{O}(N^3)$.

The CAD nodes are typically distributed among available engineering workstations. They could also be executed on a single parallel machine or the compute engine itself. The individual nodes are fully independent. Hence, the system is tolerant of node crashes and it is easy to add or delete nodes. The nodes are "greedy", that is, they compete for geometry requests by checking the CAD repository. In order to avoid race conditions between nodes for the same geometry request, a node must first acquire a lock on the CAD repository. Once a lock is obtained, the node searches for the oldest geometry request and releases the lock. This process is further complicated by the fact that all communications between the node, the CAD-request repository, the part storage location, and the compute engines are performed using secure-shell commands (see Fig. 4). Once a geometry request is processed, the node notifies the optimization process that the surface triangulation is ready. The surface triangulation is pulled from the node by the optimization process when the analysis of that particular configuration is required. This facilitates the downloading of the surface triangulations in parallel.

9 Results and Discussion

Two design examples are presented to investigate the effectiveness of the new optimization framework. We compare the genetic and BFGS quasi-Newton algorithms in both examples. All geometry models are constructed using the Pro/ENGINEER CAD system. In the first example, we address noise in the optimization process. By the use of a simple "2-D" geometry, the contribution of noise due to changes in the cut-cells of the volume mesh is isolated. A more complex geometry is used for the second example. This example focuses on the efficiency of the CAD/CAPRI module and the interface with the optimization procedure.

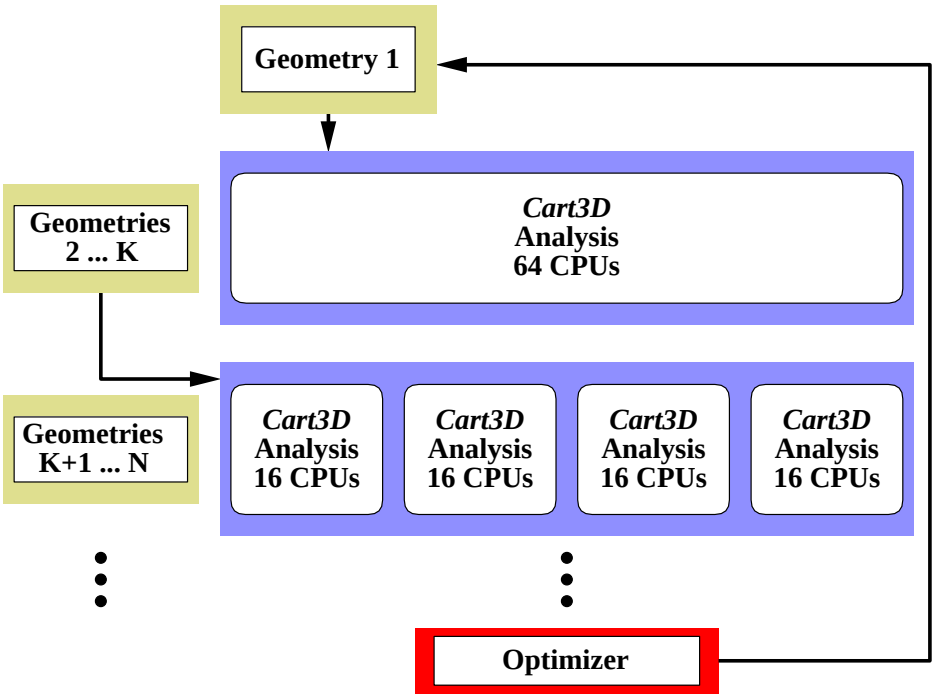


Figure 5: Dynamic allocation of processors to mask the latency of CAD geometry processing (based on 64 CPUs as an example)

9.1 2-D Design Example

The first design example is based on a two-dimensional transonic flow over the NACA 0012 airfoil. The freestream Mach number is 0.7 and the initial angle of attack is 3 deg. The airfoil section is actually modeled as a three-dimensional wing of unit-span, with no twist and no taper, as shown at the bottom of Fig. 2. All shape perturbations are performed by the CAD/CAPRI geometry module.

The following experiment is performed to estimate the level of discretization noise in a typical design landscape. We hold the airfoil geometry and flow conditions fixed, and we monitor the variation in the lift and drag coefficients due to rigid body motion of the airfoil. Similar experiments have been reported by Anderson *et al.* [33] for unstructured meshes and Dadone *et al.* [13] for Cartesian meshes. Ideally, the aerodynamic coefficients should remain constant. However, the changes in the cut-cells, and the corresponding changes in the truncation error of the spatial discretization, introduce a variation, or noise, in the aerodynamic coefficients that should be minimized to ensure smooth design landscapes.

The extent of rigid body motion is based on the coarsest cell on the body of the airfoil, which is roughly $0.7\%c$ for a mesh with $20,576$ cells on the symmetry plane after 14 levels of cell refinement. The cell is traversed in 20% increments in both the horizontal and vertical directions. Note that for this relatively simple transonic flow, the flow solver converges at least six orders of magnitude. Hence, the variation in the lift and drag coefficients is primarily influenced by the local mesh truncation error. The following summary provides guidelines that minimize the sensitivity of the aerodynamic coefficients to the mesh:

- Sub-cell information [9] regarding the variation of the surface within the cut cell should be used.
- Additional local refinement of sharp features, such as trailing edges, should be performed. The mesh generator has been modified to perform this task automatically.

The resulting peak-to-peak variation in lift is limited to 0.5%, while the variation in drag is 1.7% or roughly 2 counts as the airfoil traverses the mesh. The noise is the truncation error of the Cartesian cells projected into the functionals of interest. This is an indication of how close an optimization algorithm can approach the optimal solution. In poorly-scaled, or flat, regions of the design space, a gradient method may stall due to the presence of such noise. However, once the level of noise is established, we use this information to select a sufficiently large finite-difference gradient stepsize [34] to maximize the performance of the gradient method for the given level of mesh refinement.

For design problems that involve only local shape changes, the variation of the functionals is smaller. Further noise reductions are obtained by the use of the right-

triangle tessellation algorithm (see Fig. 3). The quality-based triangulation is more sensitive to small shape perturbations, which results in local, non-smooth changes in the surface discretization and may trigger changes in the refinement boundaries of the volume mesh. Overall, the issue of noise remains a subject of ongoing research, with present focus on limiter formulations in the cut-cells.

We demonstrate the performance of the framework on a lift-constrained drag minimization problem. The objective function is given by

$$\mathcal{J} = \begin{cases} \omega_{\mathrm{L}} \left(1 - \frac{C_{\mathrm{L}}}{C_{\mathrm{L}}^*}\right)^2 + \omega_{\mathrm{D}} \left(1 - \frac{C_{\mathrm{D}}}{C_{\mathrm{D}}^*}\right)^2 & \text{if } C_{\mathrm{D}} > C_{\mathrm{D}}^* \\ \omega_{\mathrm{L}} \left(1 - \frac{C_{\mathrm{L}}}{C_{\mathrm{L}}^*}\right)^2 & \text{otherwise} \end{cases} \tag{4}$$

where C_{D}^* and C_{L}^* represent the target drag and lift coefficients, respectively. The target lift coefficient is set to 0.545, which is the lift coefficient for the initial shape and flow conditions, and the target drag coefficient is set to 0.002, which represents a five-fold reduction in drag from the initial conditions. The weights ω_{L} and ω_{D} are user specified constants set to 1.0 and 0.005, respectively. The angle of attack and the vertical position of two B-spline control points on the upper surface of the airfoil are used as design variables (see Fig. 2).

Figure 6 shows the convergence of the objective function for both the BFGS quasi-Newton (denoted as gradient) and genetic (denoted as GA) algorithms. The label "Design Iterations" in Fig. 6 refers to the number of generations evaluated by the genetic algorithm, and the number of objective function and gradient evaluations performed by the quasi-Newton algorithm. We use 16 chromosomes, i.e. objective function evaluations, to define a generation of the genetic algorithm. The quasi-Newton algorithm requires seven objective function evaluations at each design iteration. The two optimization algorithms converge to the same solution. The L_2-norm of the gradient vector is reduced by 2.5 orders of magnitude. Assuming that the objective function is converged within 15 design iterations for both optimizers, the quasi-Newton algorithm required 105 function evaluations, while the genetic algorithm required 240 function evaluations.

Figure 7 shows the convergence of the lift and drag coefficients for the quasi-Newton algorithm. Note that the drag coefficient is reduced by at least a factor of two when compared with the initial design. Figure 8 shows the initial and final pressure distributions and airfoil shapes.

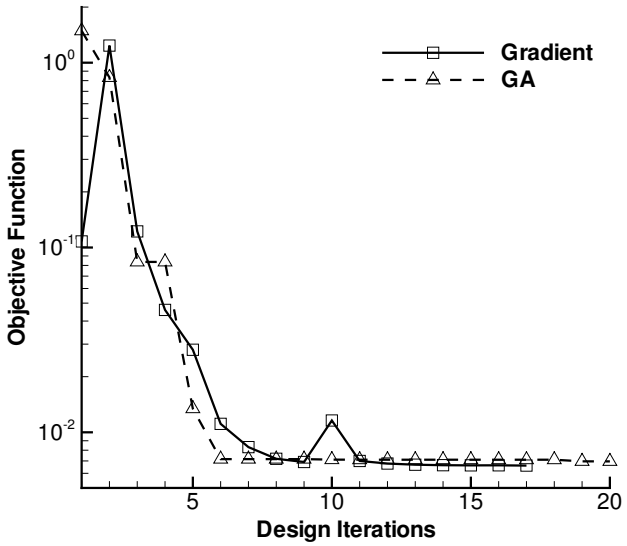


Figure 6: Objective function convergence history for the lift-constrained drag minimization problem (3 design variables)

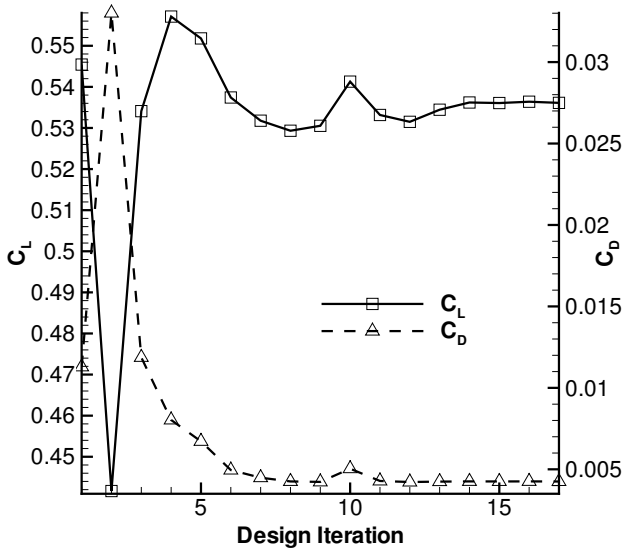


Figure 7: Convergence of the lift and drag coefficients for the quasi-Newton algorithm (3 design variables)

9.2 3-D Design Example

The second design example is based on the configuration shown in Fig. 9. This generic model is a CAD assembly of five parts consisting of a fuselage with a bluff base, a wing, a canard, a canted tail, and an engine cluster. The wing and canard are constructed from the same CAD model, which was also used in the first design example and is shown in Fig. 2. At the assembly level, the wing and canard parts are "attached" to the fuselage via two parameters, their horizontal and vertical locations, respectively. These parameters are constrained to intersect the projection of the fuselage on the symmetry plane within the CAD system. This simple construct avoids non-physical configurations, for example wings that detach from the fuselage during the optimization, even

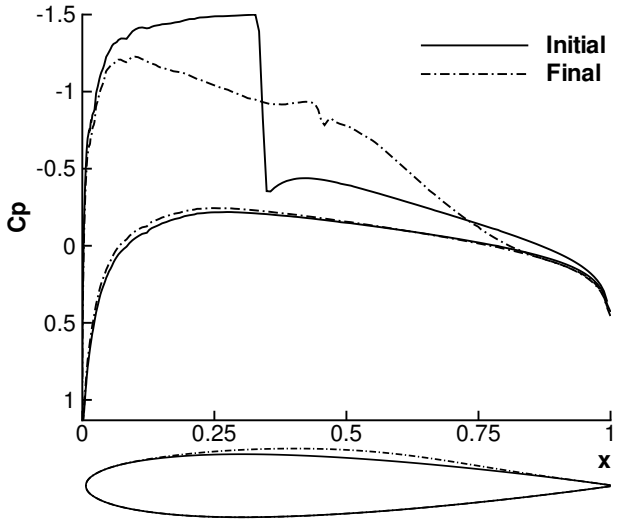


Figure 8: Pressure distribution and airfoil shapes for the lift-constrained drag minimization problem (3 design variables)

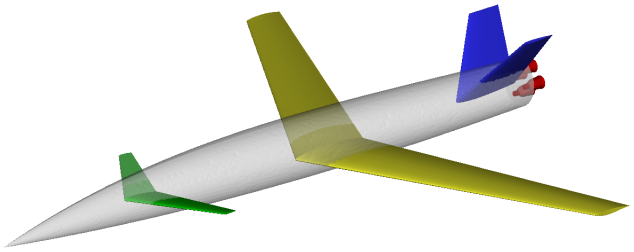


Figure 9: Model configuration for the second design example (before component intersection)

if the fuselage shape and dimensions change.

Before presenting optimization results, we characterize the performance of the optimization framework. We focus on the analysis module, see Fig 1, as this is the most expensive part of the framework. Table 1 presents average CPU timing results for the CAD model regeneration and surface triangulation using CAPRI. The timings for the fuselage and wing parts are representative of any other component in the assembly. The CAD-model regeneration times are slightly faster for changes that do not require shape modifications, i.e. no profile section changes. It is clear from Table 1 that CAD-model regeneration times are not a significant expense even for problems with many design variables.

The CPU time for surface triangulation is greatly influenced by the choice of the triangulation algorithm. For the fuselage, CAPRI uses the quality-based triangulation algorithm. This is in contrast to the wing surfaces, where the right-triangle tessellation algorithm is used. To further elucidate the performance reported in Table 1, the quality-based triangulation algorithm generates roughly 500 triangles per CPU sec., while the right-triangle tessellation algorithm generates roughly $3,300$ triangles per CPU sec. While the time required for surface triangulation is not prohibitive, it is important to avoid all unnecessary re-triangulations during the optimization. This is accomplished by caching an associated baseline triangulation for each part prior to the optimization and tracking parameter changes. For example, we tag design variables that control relative motion between components, since a change in these parameters does not require surface re-triangulation.

Table 2: Wallclock times for individual components of the *Cart3D* module (600 MHz R14000 SGI Origin 3000)

Component	Time (s)	Algorithm
Mesh Generation[a]	132.0	Serial
Flow Solution[b]	455.0	Parallel
Mesh Solution Transfer	26.0	Serial

[a] Includes component intersection (definition of wetted surface), mesh generation, flow-solver domain decomposition, and multigrid coarse-mesh generation

[b] Using 64 processors

Table 2 presents average timing results for individual components within the *Cart3D* analysis module. The volume mesh contains roughly 1.5 million cells for a half-span model of the configuration and 64 processors are used to obtain the flow solution. The time for the mesh-solution transfer algorithm used to "warm-start" finite-difference gradient computations is also shown.

Valuable information regarding the CPU efficiency during a design iteration is obtained by comparing Tables 1 and 2. For example, suppose that we have only one CAD license available and that the design problem of interest involves design variables associated with both the fuselage and wing. Then, the timings in Tables 1 and 2 indicate that the time required to complete a CAD-model regeneration and surface triangulation is a factor of six smaller than the time required for a flow solution. This means that by the time the analysis module completes the flow solution of the first chromosome of the GA or the base-state of the gradient method, six new surface triangulations are ready for analysis. By subdividing the available CPUs of the optimization process, we execute multiple analysis modules in parallel to enhance the parallel efficiency of the optimization framework.

We consider the optimization problem of attaining a nearly zero pitching moment coefficient for the configuration shown in Fig. 9 by optimizing the canard control surface. The lift coefficient is constrained by the initial lift of the configuration. The design variables are the control surface aspect ratio, twist, and position along the center line of the fuselage. The problem has two local optima, the tail or

Table 1: Average CPU time for CAD-model regeneration and tessellation (600 MHz R14000 SGI Octane Workstation, Pro/ENGINEER kernel)

Part	CAD-Model Regeneration (s)	Tessellation (s)	Number of Triangles	Tessellation Algorithm
Fuselage	2.0[a]	93.3	$\approx 41,000$	Quality-based
Wing	3.0[b]	16.5	$\approx 50,000$	Right-triangle

[a] No shape-section change, only global parameter modifications
[b] Shape-section change and planform parameter modifications

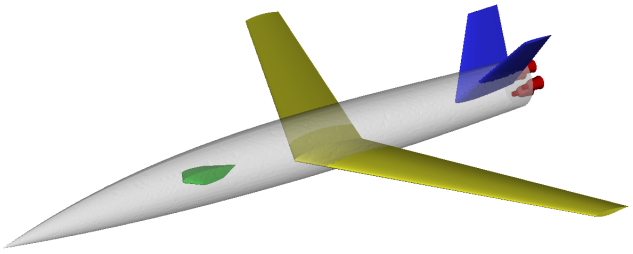


Figure 10: Example configuration where the control surface is not part of the wetted surface

canard configuration, with the canard configuration as the global optimum due to an aft location of the center of gravity. For optimization using the genetic algorithm, the canard area is also a design variable. This introduces the possibility of a topology change in the design space, since the resulting wetted surface may not include a control surface, as shown in Fig. 10. We use 16 chromosomes for each generation of the genetic algorithm. For the gradient-based quasi-Newton algorithm, the control surface area is kept constant and we start from a canard configuration, i.e. the control surface is positioned in front of the center of gravity. The freestream Mach number is 0.85 and the angle of attack is 1.0 deg.

The objective function is similar to Eq. 4, with a target lift coefficient of 0.222 and a target pitching moment coefficient of 0.001. The initial pitching moment is -0.0714. Figure 11 shows the convergence history of the objective function. Note that the label "Design Iteration" refers to the number of generations evaluated by the genetic algorithm, and the number of objective function and gradient evaluations by the quasi-Newton algorithm. Both optimization methods trim the configuration at the given flight conditions. The gradient has been reduced by almost three orders of magnitude. The genetic algorithm converges within six design iterations, requiring only 96 function evaluations. The quasi-Newton algorithm requires 56 function evaluations.

Figures 12(a) and 12(b) show the initial and final designs for the quasi-Newton algorithm. The control surface converged to the minimum allowable forward location on the fuselage (8% of fuselage length), a twist angle of 2.98 deg., and an aspect ratio of 6.03. Note that the control surface area

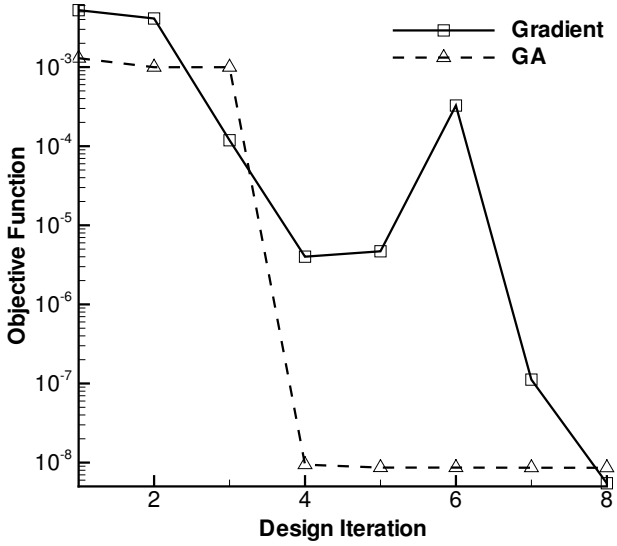
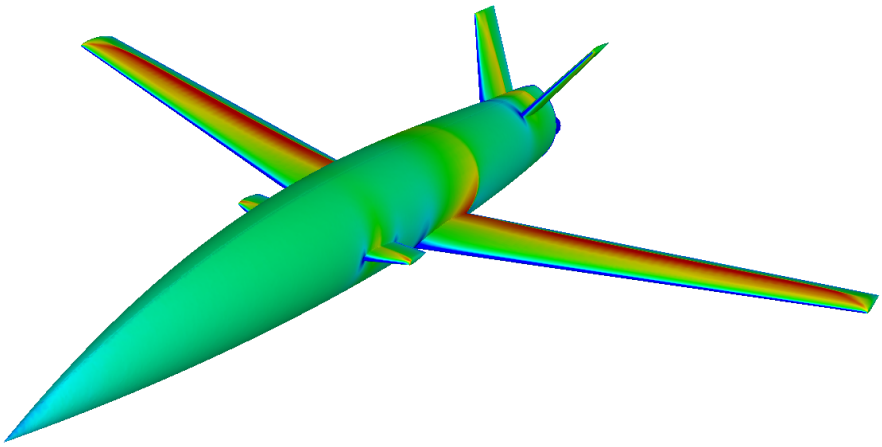


Figure 11: Objective function convergence

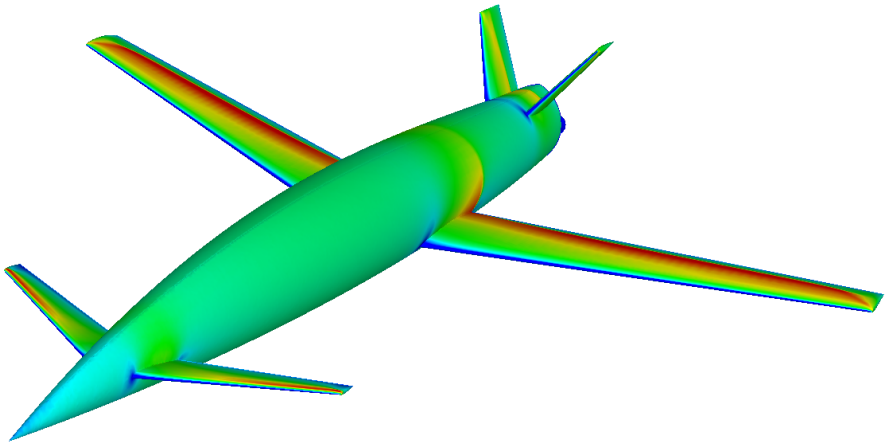
is fixed at 60.0 during the optimization. Figure 12(c) shows the final design using the genetic algorithm. For this case, the optimization converged to the upper bound of the control surface area, which is 60.0, a forward location of 8.2% of fuselage length, a twist angle of 3.41 deg., and an aspect ratio of 4.36. The difference in the two designs indicates that the optimization problem does not have a unique solution. There may be many control surfaces that trim this configuration and further constraints are required to define a unique problem.

10 Conclusions and Future Work

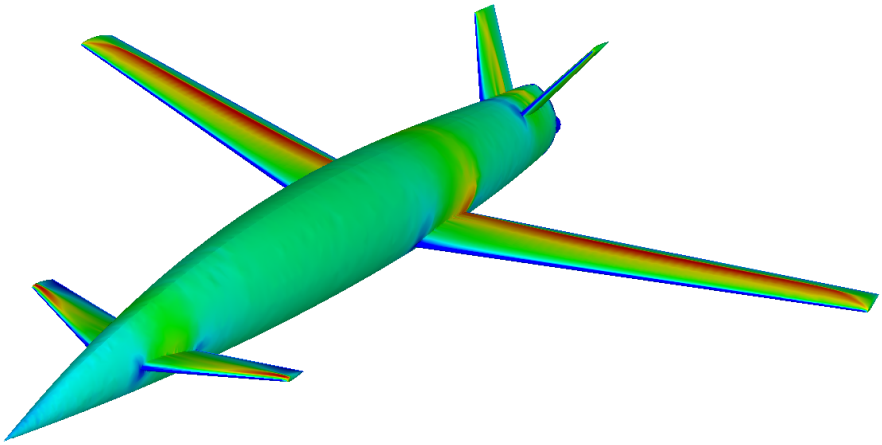
An automated optimization framework has been developed for inviscid-flow aerodynamic design problems. Key aspects of the framework include the use of a robust and efficient Cartesian method, a direct interface to a feature-based CAD system, and the use of two optimization algorithms, namely a quasi-Newton and genetic algorithms. The CAD-system interface provided by CAPRI, which controls geometry regeneration and surface tessellation tasks, performed well for the selected design examples. Two major advan-



(a) Initial configuration for quasi-Newton algorithm



(b) Final configuration, quasi-Newton algorithm



(c) Final configuration, genetic algorithm

Figure 12: Surface Mach number ($M_\infty = 0.85$, $\alpha = 1°$). Mach numbers above 1.3 are red and Mach numbers below 0.5 are blue.

tages of the Cartesian method have been demonstrated: 1) the decoupling of the surface mesh form the volume mesh allows the direct use of surface tessellations generated by CAPRI, regardless of topology or large shape changes, and 2) the component-based approach of *Cart3D* alleviates the demands on the CAD system and significantly reduces surface tessellation tasks by reusing cached component triangulations.

We have shown that the level of noise in the design landscape can be reduced to levels acceptable for gradient-based algorithms. As a result, both optimizers performed well for the selected design problems. Although the gradient-based algorithm requires less function evaluations for the examples presented, we found the genetic algorithm more tolerant of design landscape noise, which permits the use of coarser meshes. We plan to investigate this further in our future work. In addition, we intend to apply the present framework to more difficult optimization problems and real-life geometries. Such problems motivate the use of hybrid strategies and more sophisticated optimization methods.

11 Acknowledgments

The authors gratefully acknowledge Robert Haimes (MIT) for his assistance with CAPRI, in particular the implementation of the new tessellation algorithm and master-model improvements. The authors would also like to thank Peter Gage (NASA Ames), Alexander Te (NASA Ames), and Curran Crawford (MIT) for their help with parametric geometry models. This work was performed while the first author held a National Research Council Research Associateship Award at the NASA Ames Research Center.

References

[1] Haimes, R. and Follen, G., "Computational Analysis PRogramming Interface," *Proceedings of the 6th International Conference on Numerical Grid Generation in Computational Field Simulations*, edited by Cross, Eiseman, Hauser, Soni, and Thompson, University of Greenwich, 1998.

[2] Aftosmis, M. J., Delanaye, M., and Haimes, R., "Automatic Generation of CFD-Ready Surface Triangulations from CAD Geometry," AIAA Paper 99–0776, Reno, NV, Jan. 1999.

[3] Haimes, R. and Aftosmis, M. J., "On Generating High Quality "Water-tight" Triangulations Directly from CAD," Tech. rep., Meeting of the International Society for Grid Generation, (ISGG), Honolulu, HI, June 2002.

[4] Alonso, J. J., Martins, J. R. R. A., Reuther, J. J., Haimes, R., and Crawford, C., "High-Fidelity Aero-Structural Design Using a Parametric CAD-Based Model," AIAA Paper 2003–3429, Orlando, FL, June 2003.

[5] Haimes, R. and Crawford, C., "Unified Geometry Access for Analysis and Design," Tech. rep., 12th International Meshing Roundtable, Santa Fe, NM, Sept. 2003.

[6] Reuther, J. J., Jameson, A., Alonso, J. J., Rimlinger, M. J., and Saunders, D., "Constrained Multipoint Aerodynamic Shape Optimization Using an Adjoint Formulation and Parallel Computers, Part 1," *Journal of Aircraft*, Vol. 36, No. 1, 1999, pp. 51–60.

[7] Nielsen, E. J. and Anderson, W. K., "Recent Improvements in Aerodynamic Design Optimization on Unstructured Meshes," *AIAA Journal*, Vol. 40, No. 6, 2002, pp. 1155–1163.

[8] Cliff, S. E., Thomas, S. D., Baker, T. J., and Jameson, A., "Aerodynamic Shape Optimization Using Unstructured Grid Methods," AIAA Paper 2002–5550, Atlanta, GA, Sept. 2002.

[9] Aftosmis, M. J., "Solution Adaptive Cartesian Grid Methods for Aerodynamic Flows with Complex Geometries," Lecture notes, von Karman Institute for Fluid Dynamics, Series: 1997-02, Brussels, Belgium, March 1997.

[10] Aftosmis, M. J., Berger, M. J., and Melton, J. E., "Robust and Efficient Cartesian Mesh Generation for Componenet-Based Geometry," *AIAA Journal*, Vol. 36, No. 6, 1998, pp. 952–960.

[11] Murman, S. M., Aftosmis, M. J., and Berger, M. J., "Implicit Approaches for Moving Boundaries in a 3-D Cartesian Method," AIAA Paper 2003–1119, Reno, NV, Jan. 2003.

[12] Rodriguez, D. L., "Response Surface Based Optimization With a Cartesian CFD Method," AIAA Paper 2003–0465, Jan. 2003.

[13] Dadone, A. and Grossman, B., "Efficient Fluid Dynamic Design Optimization Using Cartesian Grids," AIAA Paper 2003–3959, Orlando, FL, June 2003.

[14] Obayashi, S., "Aerodynamic Optimization with Evolutionary Algorithms," *Inverse Design and Optimization Methods, Lecture Series 1997-05*, edited by R. A. Van den Braembussche and M. Manna, von Karman Institute for Fluid Dynamics, Brussels, Belgium, 1997.

[15] Marco, N., Désidéri, J.-A., and Lanteri, S., "Multi-Objective Optimization in CFD by Genetic Algorithms," Tech. Rep. 3686, Institut National De Recherche En Informatique Et En Automatique (INRIA), France, April 1999, Also see www.lania.mx/∼ccoello/EMOO/EMOObib.html.

[16] Holst, T. L. and Pulliam, T. H., "Aerodynamic Shape Optimization Using a Real-Number-Encoded Genetic Algorithm," AIAA Paper 2001–2473, Anaheim, CA, June 2001.

[17] Dennis Jr., J. E. and Schnabel, R. B., *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice-Hall, Englewood Cliffs, N.J., 1983.

[18] Jameson, A., "Aerodynamic Shape Optimization Using the Adjoint Method," Lecture notes, von Karman Institute for Fluid Dynamics, Brussels, Belgium, Feb. 2003.

[19] Elliott, J. and Peraire, J., "Constrained, Multipoint Shape Optimisation for Complex 3D Configurations," *Aeronautical Journal*, Vol. 102, No. 1017, 1998, pp. 365–376.

[20] Nemec, M. and Zingg, D. W., "Newton–Krylov Algorithm for Aerodynamic Design Using the Navier–Stokes Equations," *AIAA Journal*, Vol. 40, No. 6, 2002, pp. 1146–1154.

[21] Torczon, V., "On the Convergence of Pattern Search Algorithms," *SIAM Journal on Optimization*, Vol. 7, No. 1, 1997, pp. 1–25.

[22] Alexandrov, N. M., Nielsen, E. J., Lewis, R. M., and Anderson, W. K., "First-Order Model Management with Variable-Fidelity Physics Applied to Multi-Element Airfoil Optimization," AIAA Paper 2000–4886, September 2000.

[23] Ong, Y. S., Nair, P. B., and Keane, A. J., "Evolutionary Optimization of Computationally Expensive Problems via Surrogate Modeling," *AIAA Journal*, Vol. 41, No. 4, 2003, pp. 687–696.

[24] Aftosmis, M. J., Berger, M. J., and Adomavicius, G., "A Parallel Multilevel Method for Adaptively Refined Cartesian Grids with Embedded Boundaries," AIAA Paper 2000–0808, Reno, NV, Jan. 2000.

[25] Charlton, E. F., *An Octree Solution to Conservation-laws over Arbitrary Regions (OSCAR) with Applications to Aircraft Aerodynamics*, Ph.D. thesis, University of Michigan, 1997.

[26] Samareh, J. A., "Survey of Shape Parametrization Techniques for High-Fidelity Multidisciplinary Shape Optimization," *AIAA Journal*, Vol. 39, No. 5, 2001, pp. 877–883.

[27] Samareh, J. A., "Novel Multidisciplinary Shape Parametrization Approach," *Journal of Aircraft*, Vol. 38, No. 6, 2001, pp. 1015–1023.

[28] Townsend, J. C., Samareh, J. A., Weston, R. P., and Zorumski, W. E., "Integration of a CAD System Into an MDO Framework," NASA TM 1998–207672, May 1998.

[29] Moré, J. J. and Thuente, D. J., "Line Search Algorithms with Guaranteed Sufficient Decrease," *ACM Transactions on Mathematical Software*, Vol. 20, No. 3, 1994, pp. 286–307.

[30] Aftosmis, M. J., Berger, M. J., and Murman, S. M., "Applications of Space-Filling-Curves to Cartesian Methods for CFD," AIAA Paper 2004–1232, Reno, NV, Jan. 2004.

[31] Eldred, M. S., Giunta, A. A., and van Bloemen Waanders, B. G., *DAKOTA, A Multilevel Parallel Object-Oriented Framework for Design Optimization, Parameter Estimation, Uncertainty Quantification, and Sensitivity Analysis: Version 3.1 Users Manual*, Sandia National Laboratories, 2003, http://endo.sandia.gov/DAKOTA/software.html.

[32] Eldred, M. S. and Hart, W. E., "Design and Implementation of Multilevel Parallel Optimization on the Intel Teraflops," AIAA Paper 1998–4707, 1998.

[33] Anderson, W. K. and Venkatakrishnan, V., "Aerodynamic Design Optimization on Unstructured Grids with a Continuous Adjoint Formulation," *Computers & Fluids*, Vol. 28, 1999, pp. 443–480.

[34] Gill, P. E., Murray, W., Saunders, M. A., and Wright, M. H., "Computing Forward-Difference Intervals for Numerical Optimization," *SIAM Journal on Scientific and Statistical Computing*, Vol. 4, No. 2, 1983, pp. 310–321.